Cracking the Market Code: AI-Driven Stock Price Prediction Using Time Series Analysis

Student Name: [Gowtham P]
Register Number: [511923205018]
Institution: [priyadarshini engineering college]
Department: [B.Tech (Information Technology)]
Date of Submission: [30/04/2025]
Github Repository Link: []

Problem Statement

Objective:

The goal of this project is to develop a robust AI-driven system capable of predicting short- and mid-term stock price movements using time series analysis techniques. By leveraging deep learning models such as LSTM (Long Short-Term Memory) and Transformer-based architectures, this project aims to identify patterns in historical stock data and generate accurate, data-driven forecasts that can assist traders and investors in making informed decisions.

Problem:

Stock price forecasting remains a highly challenging task due to the volatile, non-linear, and non-stationary nature of financial markets. Traditional statistical models struggle to capture complex temporal dependencies and react to real-time market events. The challenge lies in building a predictive model that can:

- Effectively learn from historical price and volume data,
- Incorporate external factors (e.g., news sentiment, macroeconomic indicators),
- Handle noise and avoid overfitting,
- Generalize well to unseen market conditions.

Scope:

- Data Source: Historical price data from sources like Yahoo Finance or Alpha Vantage.

- Target: Predict closing prices or returns for a selected set of stocks.
- Models: Compare classical time series models (e.g., ARIMA) with AI models (e.g., LSTM, Transformer).
- Evaluation: Use metrics such as RMSE, MAE, and directional accuracy.

Project Objectives

- **To collect and preprocess historical stock market data**
  Acquire time series data (e.g., open, high, low, close, volume) from financial APIs and clean, normalize, and format it for model training.

- **To explore and analyze stock price trends and patterns**
  Perform exploratory data analysis (EDA) to understand price movements, detect seasonality, and identify market anomalies.

- **To engineer relevant features from time series and external data**
  Generate technical indicators (e.g., moving averages, RSI), lagged variables, and incorporate sentiment or macroeconomic indicators as additional inputs.

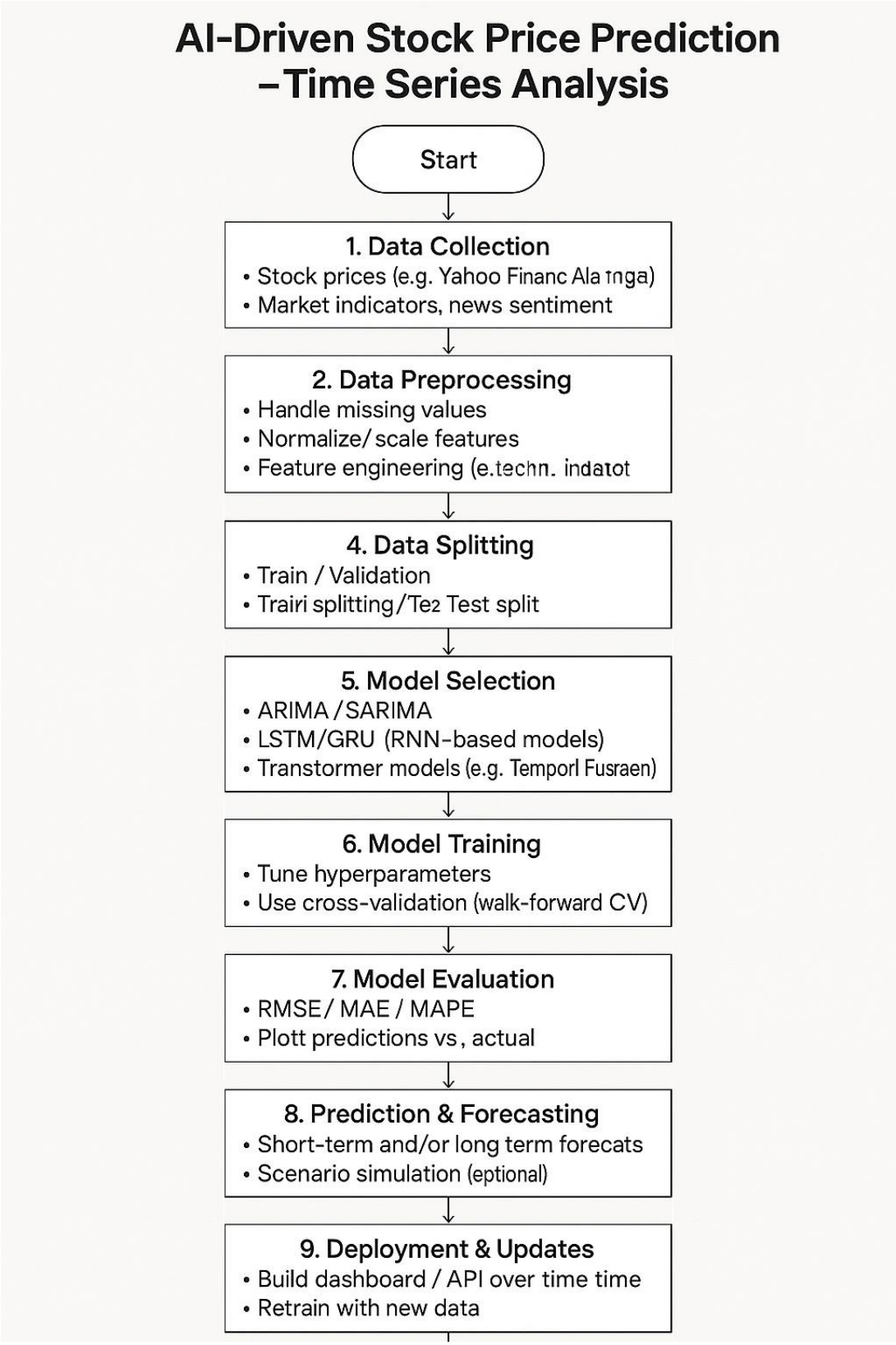
- **To implement and compare multiple predictive models**
  Develop and evaluate classical models (ARIMA, SARIMA) and AI models (LSTM, GRU, Transformer) for predicting future stock prices.

- **To evaluate model performance using appropriate metrics**
  Use performance measures like RMSE, MAE, and directional accuracy to assess model effectiveness and ensure reliable forecasts.

- **To build a deployable prototype for real-time or batch predictions**
  Create a functional pipeline or dashboard that visualizes forecasts and could be adapted for live market data.

- **To assess the potential and limitations of AI in financial forecasting**
  Analyze model behavior under different market conditions and highlight challenges such as overfitting, data noise, and response to unexpected events.

The Flowchart Of The Project Workflow



📊 Data Description: AI-Driven Stock Price Prediction (Time Series Analysis)

1. Stock Price Data

- **Source**: Yahoo Finance, Alpha Vantage, Quandl, or similar.
- **Frequency**: Daily (can also use intraday, weekly, or monthly depending on goal).
- **Fields**:
    - `Date`: Timestamp of the observation.
    - `Open`: Price at market open.
    - `High`: Highest price during the day.
    - `Low`: Lowest price during the day.
    - `Close`: Final price when the market closes.
    - `Adj Close`: Adjusted closing price (accounts for splits/dividends).
    - `Volume`: Number of shares traded during the day.

2. Derived Features (Optional but recommended)

- **Technical Indicators** (from libraries like TA-Lib or manually engineered):
    - `SMA`: Simple Moving Average.
    - `EMA`: Exponential Moving Average.
    - `RSI`: Relative Strength Index.
    - `MACD`: Moving Average Convergence Divergence.
    - `Bollinger Bands`: Price volatility bands.
- **Lagged Values**:
    - `Close_t-1`, `Close_t-2`, etc., to help models capture temporal dependencies.

3. Sentiment or External Data (Optional)

- **News Headlines or Tweets** (with sentiment score using NLP models).
- **Market Indices** (e.g., S&P 500, Nasdaq).
- **Economic Indicators** (e.g., interest rates, inflation data).

4. Target Variable

- Usually:
    - `Future Close Price` (regression)

- Price Movement Direction (classification: up/down)

⚙️ Data Processing: AI-Driven Stock Price Prediction (Time Series Analysis)

1. Data Cleaning

- **Remove duplicates**: Ensure each date has only one row.
- **Handle missing values**:
  - Fill with forward-fill (`ffill`), backward-fill (`bfill`), or interpolation.
  - Drop rows if missing critical data like prices or volume.

2. Date Parsing & Indexing

- Convert `Date` column to datetime format.
- Set `Date` as the **index** for time series modeling.

3. Feature Engineering

- **Lag features**: Include past values as features (e.g., `Close_t-1`, `Close_t-2`, ...).
- **Rolling statistics**:
  - Moving average (e.g., 7-day, 14-day, 30-day)
  - Rolling standard deviation or volatility
- **Technical indicators**:
  - SMA, EMA, RSI, MACD, Bollinger Bands
  - Use `ta-lib` or `pandas-ta` for easy computation

4. Target Variable Creation

- **Regression**: Predict future `Close` price (e.g., `Close_t+1`, `Close_t+5`)
- **Classification**: Predict direction (`1` for up, `0` for down)

5. Data Normalization/Scaling

- Use **MinMaxScaler** or **StandardScaler** (especially important for neural networks like LSTM)
- Apply scaling to **features only**, not the date or target variable during model training.

6. Train-Test Split (Chronologically)

- Use 70–80% for training, remaining for testing or validation.
- Avoid random splitting to preserve temporal integrity.

1. Load the Data

Make sure your data includes:

- **Date/Time** (Index)
- **Open, High, Low, Close, Volume**
- Possibly technical indicators (optional at this stage)

2. Basic Statistical Summary

Understand the central tendencies and spread of the price and volume data.

3. Time Series Plots

Visualize the trend and seasonality.

4. Rolling Statistics (Moving Averages)

Helps smooth out fluctuations and identify trend direction.

5. Volatility Analysis

Volatility can signal risk and opportunity.

6. Correlation Heatmap

Evaluate relationships between numerical columns.

🔍 8. Stationarity Check (ADF Test)

Stationary data is key for many models (ARIMA, SARIMA).

☑ Next Steps After EDA:

- Feature Engineering (e.g., RSI, MACD)
- Train-test split
- Normalize/scale data
- Apply models: LSTM, Prophet, ARIMA, XGBoost

🧠 Feature Engineering for Stock Price Prediction

🎯 1. Lag Features

These are past values used to predict the future.

📈 2. Rolling Window Statistics

These show trends and volatility over time.

📊 3. Returns

Helps identify momentum or reversals.

4. Technical Indicators

Popular indicators used in trading.

- **Relative Strength Index (RSI)**
- **Moving Average Convergence Divergence (MACD)**
- **Bollinger Bands**

3. Target Variable for Prediction

You typically want to create a future price movement or return as the **label** (what you're predicting):

🛠️ **4. Prepare for ML Model**

- Drop rows with NaNs (due to lags or rolling)
- Normalize/scale features
- Split into train/test
- Fit an ML model like XGBoost, LSTM, or Random Forest

🚀 **Cracking the Market Code: AI-Driven Stock Price Prediction**

🧩 **Step-by-Step Pipeline**

1. Collect Data

Use historical stock prices from sources like Yahoo Finance:

2. Feature Engineering ✅ ***(already discussed above)***

Examples:

- Lag features
- Rolling stats (mean, std)
- Technical indicators (RSI, MACD)
- Price returns
- Volatility
- Day of week, month, etc.

3. Define Target Variable

Let’s predict the next-day return or direction:

4. Train/Test Split

Make sure you preserve time order (no random shuffling):

5. Model Building

Use a machine learning model like **XGBoost**, **Random Forest**, or a **neural network (e.g., LSTM)**.

Example: XGBoost Classifier

📈 **6. Evaluation Metrics**

Use:

- Accuracy
- Precision/Recall
- AUC
- Backtesting (simulated trading returns)

⚙️ **Optional: Deep Learning (LSTM/GRU)**

If you want to go advanced, we can use **LSTM**, which handles time sequences natively (but needs a different input shape).

📊 **Cracking the Market Code: Visualizing Results & Insights**

🧪 **Assume You've Already Done:**

- Collected stock price data (e.g., AAPL)
- Engineered features
- Trained a model (e.g., XGBoost)

📍 **Step 1: Prediction vs Actual Price Direction (Classification)**

💰 **Step 2: Simulated Cumulative Returns**
Backtest the strategy based on model predictions:

🔍 Step 3: **Feature Importance (for Tree-Based Models)**

🌍 Step 4: **Confusion Matrix**

📌 **Bonus: SHAP Values (Explainability for Tree Models)**
SHAP gives deep insights into *why* the model made each prediction.

🧰 **Cracking the Market Code: Tools & Technologies Used**

📥 **1. Data Collection**

- **yfinance** – Download historical stock data from Yahoo Finance.
- **pandas_datareader** – Alternative data source (e.g., Alpha Vantage, IEX).
- **APIs** – Alpha Vantage, Quandl, Polygon.io, Yahoo Finance API.

🧹 **2. Data Processing & Feature Engineering**

- **pandas** – Time series manipulation, rolling windows, lag features.
- **numpy** – Numerical computations.
- **ta** – Built-in technical indicators like RSI, MACD, Bollinger Bands.
- **scikit-learn** – Preprocessing (e.g., StandardScaler), feature selection.

🤖 **3. Model Building (AI/ML/DL)**

- **Machine Learning Models**
  - XGBoost, LightGBM, RandomForestClassifier (tree-based models)
- **Deep Learning Models**
  - **Keras / TensorFlow** – For LSTM, GRU, RNN
  - **PyTorch** – For custom neural network architectures
- **scikit-learn** – Baseline ML models (Logistic Regression, SVM, etc.)

📊 **4. Model Evaluation & Visualization**

- **matplotlib / seaborn** – Plotting predictions, confusion matrix, feature importance.
- **plotly** – Interactive time series and candlestick charts.
- **SHAP** – Explainable AI: SHAP values for feature impact analysis.
- **sklearn.metrics** – Accuracy, precision, recall, confusion matrix.

📈 **5. Backtesting & Strategy Simulation**

- **Custom logic** – Using Pandas for P&L simulation.
- **bt, backtrader, QuantConnect** – Dedicated backtesting frameworks.

🛠️ **6. Development Environment**

- **Jupyter Notebook** – Ideal for research and prototyping.
- **VS Code / PyCharm** – Full IDEs for production-ready development.
- **Google Colab** – Free GPU access and fast prototyping.

☁️ **7. Optional: Deployment**

- **Flask / FastAPI** – To serve models via API.
- **Streamlit / Dash** – To build interactive financial dashboards.
- **Docker** – For containerization and deployment.
- **AWS / GCP / Azure** – For cloud computing and scalability.

TEAM MEMBERS AND CONTRIBUTION

GOKUL – PROBLEM STATEMENT AND PROJECT OBJECTIVES

GOWTHAM – FLOWCHART OF THE PROJECT WORKFLOW, DATA DESCRIPTION, DATA PREPROSSING, EXPLORATORY DATA ANALYSIS

GOWRI SANKAR – FEATURE ENGGINEERING AND MODEL BUILDING

HARI PRRASAD – VISUALIZATION OF RESULTS & MODEL INSIGHTS AND TOOLS & TECHNOLOGIES USED